# Galerkin Methods for Incompressible Flow Simulation

Paul Fischer

November 23, 2009

## 1 Introduction

These notes provide a brief introduction to Galerkin projection methods for numerical solution of the incompressible Navier-Stokes equations

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} &= -\nabla p + \frac{1}{Re}\nabla^2\mathbf{u} + \mathbf{f} \\ \nabla\cdot\mathbf{u} &= 0 \end{aligned} \tag{1}$$

where $\mathbf{u}$ is the velocity field, $p$ the pressure, and $Re = UL/\nu$ the Reynolds number based on characteristic velocity and length scales, $U$ and $L$, respectively. In addition to (1), we will consider subsets of the Navier-Stokes equations, in particular, the Poisson equation,

$$-\nabla^2 u = f \text{ on } \Omega, \tag{2}$$

and the unsteady convection-diffusion equation with Peclet number $Pe := UL/\alpha$ and source $q$,

$$\frac{\partial u}{\partial t} + \mathbf{c}\cdot\nabla u = \frac{1}{Pe}\nabla^2 u + q \text{ on } \Omega, \tag{3}$$

where $Pe = UL/\alpha$ is the Peclet number involving the thermal diffusivity. In all cases, the solution is to be found on the computational domain $\Omega$ with appropriate initial conditions and boundary conditions on $\partial\Omega$. For the convection-diffusion equation, we further assume $\nabla\cdot\mathbf{c} = 0$.

## 2 Time Stepping

While our main focus is on spatial discretization, we make a quick detour to timestepping to provide context as we proceed with the development of the Galerkin method. For unsteady problems, one has the option of discretizing in space first to arrive at a coupled set of $n$ ordinary differential equations (ODEs) or to discretize in time first. The choice is largely a matter of convenience but real differences can arise from the fundamental fact that discrete differential operators (matrices) typically do not commute whereas the continuous operators do.

The overall timestepping strategy can have a profound influence on the design of linear or nonlinear solvers and on the choice of spatial discretization for the given problem. To illustrate some of the considerations, we assume that we have discretized (3) in space to arrive at the following system

$$B\frac{d\underline{u}}{dt} + C\underline{u} = -\frac{1}{Pe}A\underline{u} + B\underline{q}. \tag{4}$$

Here, $\underline{u}(t) = (u_1, \ldots, u_n)^T$ is the vector of unknown basis coefficients representing $u(\mathbf{x}, t)$, $C$ the discrete convection operator, and $A$ the (negative) discrete Poisson operator, $B$ is the mass matrix. It is often desirable to have $A$ and $C$ preserve symmetries intrinsic to their continuous counterparts, namely, $A$ should be symmetric positive definite (positive real eigenvalues) and $C$ should be skew-symmetric with purely imaginary eigenvalues. In (4), $B$ may or may not be diagonal. For finite difference schemes $B = I$, the identity matrix, and for spectral element methods $B$ is diagonal and as such is trivially inverted. For compact finite difference schemes, standard finite element methods, and modal-based spectral methods $B$ is not diagonal but it is often possible to modify the scheme to avoid unnecessary inversion of $B$. (Note: We will use underscore throughout to indicate vectors of dimension $n \gg 1$ associated with the spatial discretization, and bold face to indicate *vector fields* in $\mathbb{R}^d$, $d$=1, 2, or 3, e.g. $\mathbf{u} = (u, v, w) = (u_1, u_2, u_3) = (u_x, u_y, u_z)$ for velocity in $\mathbb{R}^3$ and $\mathbf{x} = (x, y) = (x_1, x_2)$ for the position vector in $\mathbb{R}^2$.)

If we view (4) as a (linear) model that captures several of the essential features of the Navier-Stokes equations, then a reasonable approach to time discretization is to treat the symmetric diffusion term implicitly and the nonsymmetric convective term explicitly. Rearranging (4),

$$B\frac{d\underline{u}}{dt}\bigg|_{t^m} + \frac{1}{Pe}A\underline{u}\bigg|_{t^m} \quad = \quad -C\underline{u}|_{t^m}\,, \tag{5}$$

we approximate the derivative at time level $t^m$ by a $k$th-order backward difference, the diffusion term implicitly, and the convective term on the right with extrapolation. For $k$=2 with uniform timestep size $\Delta t$, these approximations yield

$$\frac{B\left(3\underline{u}^m - 4\underline{u}^{m-1} + \underline{u}^{m-2}\right)}{2\Delta t} + \frac{1}{Pe}A\underline{u}^m \quad = \quad -\left(2C\underline{u}^{m-1} - C\underline{u}^{m-2}\right) + O(\Delta t^2), \tag{6}$$

which can be further simplified to

$$H\underline{u}^m \quad = \quad 2B\underline{u}^{m-1} - \frac{1}{2}B\underline{u}^{m-2} - \Delta t\left(2C\underline{u}^{m-1} - C\underline{u}^{m-2}\right) + O(\Delta t^3), \tag{7}$$

where we've introduced the discrete Helmholtz operator, $H := \frac{3}{2}B + \frac{\Delta t}{Pe}A$.

The scheme (6) is globally $O(\Delta t^2)$ accurate. That is, for *fixed* final time $T$, the difference between the exact solution to (5) and the numerical approximation generated by the sequence (7) will scale as $c\Delta t^2$ for some constant $c$ as $\Delta t \longrightarrow 0$. Note that this result applies only to the *truncation error*, that is, the error that results from using the finite-difference/extrapolation (BDF/EXT) scheme. There is also the issue of *round-off error* that results from using finite precision arithmetic. The most common working precision in scientific computing is 64-bit, which yields about 15 digits of accuracy, though 32-bit ($\sim$ 7 digits) is also encountered (particularly in graphics). Round-off errors tend to accumulate with more time steps and thus one can encounter *larger* errors when $\Delta t$ is decreased too far.

The rationale for semi-implicit formulation (6) is that symmetric systems (e.g., $H$) are relatively easy to solve, whereas nonsymmetric (and, for the Navier-Stokes equations, nonlinear) systems are more difficult to solve. In addition, explicit schemes have a stability constraint that limits the timestep size to satisfy $\max_i |\lambda_i| \Delta t < c$, where $c$ is an order-unity constant and $\lambda_i$ is the eigenvalue of explicit operator. Because it represents a second-order differential operator, the maximum eigenvalue of $A$ (or, more precisely, $B^{-1}A$) typically scales as $1/\Delta x_{\min}^2$, whereas that of the first-order convection operator $C$ scales as $1/\Delta x_{\min}$. Explicit treatment of $A$ would thus imply $\Delta t = O(\Delta x^2)$ whereas the convective term imposes only a first-order constraint $\Delta t = O(\Delta x)$ and implicit diffusion is thus often more of an imperative than implicit convection.

Advancement of (6) requires the solution of $H\underline{u}^m = \underline{f}^m$ at each timestep, where $\underline{f}^m$ represents the terms on the right. Formation of $\underline{f}^m$ is relatively cheap, requiring only a single matrix-vector product for each of $C$ and $B$. For the Galerkin formulation $H$ is symmetric positive definite (SPD) and therefore relatively easy to solve using either a direct approach based on Cholesky factorization or an iterative approach using preconditioned conjugate gradients (PCG). For most 3D problems PCG is *much* faster because it does not require factorization of $H$. Instead, preconditioned iterative methods such as PCG or a nonsymmetric counterpart (e.g., GMRES), only require matrix-vector products of the form $\underline{y} = H\underline{x}$ and $\underline{y} = M\underline{x}$, where $M$ is the preconditioner. The savings stems from the fact that $H$ is generally quite sparse with only $O(n)$ nonzero entries, whereas any factorization of $H$ will typically generate at least $O(n^{\frac{4}{3}})$ nonzeros. With PCG, it is possible to solve systems with $n \approx 10^9$ in just a few seconds on parallel machines. Direct approaches are currently limited to $n \approx 10^6$. For a 3D simulation, $n = 10^9$ translates into roughly 1000 points in each direction.

For a desired tolerance, $\epsilon$, the number of PCG iterations scales as $m_{\text{iter}} \sim \sqrt{\kappa} \log \epsilon$, where $\kappa = \kappa(MH) := \mu_n/\mu_1$ is the condition number of $(MH)$ and $\mu_1$, $\mu_n$ represent respective mininum and maximum eigenvalues of the preconditioned system $MH$. In the convection dominated ($Pe \gg 1$) limit, PCG will typically converge very rapidly using a simple diagonal preconditioner, e.g., $M^{-1} = \text{diag}(H)$ or even $\text{diag}(B)$. In the latter case, the iteration count is governed by the condition number of $B^{-1}H = I + \frac{2\Delta t}{3Pe} B^{-1}A$, which will be very close to the identity matrix for small $\Delta t$ and large $Pe$. With iterative solvers, it is possible to further reduce computational effort by exploiting the fact that the solution to (6) represents the smooth evolution of a function. As a result, the solution at $t^m$ is very near the solutions at earlier times $t^{m-k}$, $k = 1, \ldots, K$, and one can use this information to generate very good approximations to $t^m$ without having to solve any system by projecting $\underline{u}^m$ onto $\text{span}\{\underline{u}^{m-1}, \underline{u}^{m-2}, \ldots, \underline{u}^{m-l}\}$ in an appropriate (computable) norm.

Further discussion of timestepping issues for the incompressible Navier-Stokes equations, including a brief overview of an unconditionally stable characteristics-based scheme, is presented in Appendix A.

# 3 Galerkin Methods

We turn now to the question of spatial discretization. We introduce the Galerkin method through the classic Poisson problem in $d$ space dimensions,

$$-\nabla^2 \tilde{u} = f \text{ on } \Omega, \qquad \tilde{u} = 0 \text{ on } \partial\Omega. \tag{8}$$

Of particular interest for purposes of introduction will be the case $d = 1$,

$$-\frac{d^2\tilde{u}}{dx^2} = f, \qquad \tilde{u}(\pm 1) = 0. \tag{9}$$

We use $\tilde{u}$ to represent the exact solution to (8) and $u$ to represent our numerical solution.

Beginning with a *finite-dimensional* approximation space $X_0^N$ and associated set of basis functions $\{\phi_1, \phi_2, \ldots, \phi_n\} \in X_0^N$ satisfying the homogeneous boundary condition $\phi_i = 0$ on $\partial\Omega$, the standard approach to deriving a Galerkin scheme is to multiply both sides of (8) by a test function $v \in X_0^N$, integrate over the domain, and seek a solution $u(\mathbf{x}) := \sum u_j \phi_j(\mathbf{x})$ satisfying

$$-\int_\Omega v\nabla^2 u \, dV = \int_\Omega v f \, dV \;\; \forall v \in X_0^N. \tag{10}$$

The Galerkin scheme is essentially a method of undetermined coefficients. One has $n$ unknown basis coefficients, $u_j$, $j = 1, \ldots, n$ and generates $n$ equations by successively choosing test functions $v$ that span $X_0^N$ (e.g., $v = \phi_i$, $i = 1, \ldots, n$). Equating both sides for every basis function in $X_0^N$ ensures that the residual, $r(\mathbf{x}; u) := -\nabla^2 u - f$ is orthogonal to $X_0^N$, which is why these methods are also referred to weighted residual techniques.

Defining the $\mathcal{L}^2$ inner product $(f, g) := \int_\Omega fg \, dV$, (10) is equivalent to finding $u \in X_0^N$ for which

$$(v, r) \;:=\; \int_\Omega v\, r(\mathbf{x},\, u)\, dV \;=\; 0, \qquad \forall v \in X_0^N. \tag{11}$$

That is, $r(u)$ is orthogonal to $v$ or, in this case, the entire space: $r \perp X_0^N$. Convergence, $u \longrightarrow \tilde{u}$, is achieved by increasing $n$, the dimension of the approximation space. As the space is completed, the only function that can be orthogonal to all other functions is the zero function, such that $u \equiv \tilde{u}$.

It is important to manipulate the integrand on the left of (10) to equilibrate the continuity requirements on $u$ and $v$. Integrating by parts, one has

$$-\int_\Omega v\nabla^2 u \, dV = \int_\Omega \nabla v \cdot \nabla u \, dV - \int_{\partial\Omega} v\nabla u \cdot \hat{\mathbf{n}} \, dA \tag{12}$$

The boundary integral vanishes because $v = 0$ on $\partial\Omega$ ($v \in X_0^N$) and the Galerkin formulation reads: *Find* $u \in X_0^N$ *such that*

$$\int_\Omega \nabla v \cdot \nabla u \, dV = \int_\Omega v f \, dV \;\; \forall v \in X_0^N. \tag{13}$$

Note that the integration by parts serves to reduce the continuity requirements on $u$. We need only find a $u$ that is once differentiable. That is, $u$ will be continuous, but $\nabla u$ need not be. Of course, if $\nabla\tilde{u}$ is continuous, then $\nabla u$ will converge to $\nabla\tilde{u}$ for a properly formulated and implemented method.

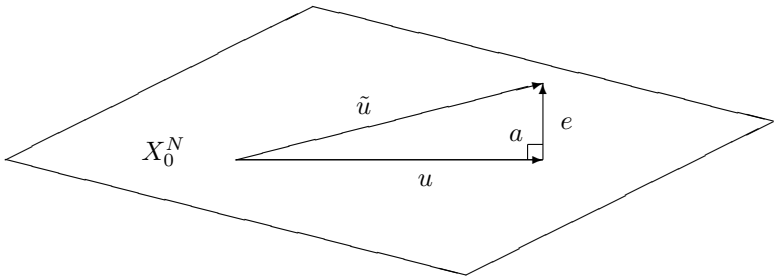


Figure 1: $a$-orthogonal projection of $\tilde{u}$ onto $X_0^N$.

Equation (13) is the point of departure for most finite element, spectral element, and spectral formulations of this problem. To leading order, these formulations differ primarily in the choice of the $\phi_i$s and the approach to computing the integrals, both of which influence the computational efficiency for a given problem. The Galerkin formulation of the Poisson problem has many interesting properties. We note a few of these here:

- We've not yet specified requisite properties of $X_0^N$, which is typically the starting point (and often the endpoint) for mathematical analysis of the finite element method. Two function spaces are of relevance to us: $\mathcal{L}^2$, which is the space of functions $v$ satisfying $\int_\Omega v^2\, dV < \infty$, and $\mathcal{H}_0^1$, which is the space of functions $v \in \mathcal{L}^2$ satisfying $\int_\Omega \nabla v \cdot \nabla v\, dV < \infty$ and $v = 0$ on $\partial\Omega$. Over $\mathcal{L}^2$, we have the inner product $(v, w) := \int_\Omega vw\, dV$ and norm $||v|| := \sqrt{(v, v)}$. For any $v, w \in \mathcal{H}_0^1$, we also define the energy inner product $a(v, w) := \int_\Omega \nabla v \cdot \nabla w\, dV$ and associated energy norm, $||w||_a := \sqrt{a(w, w,)}$. For the (continuous) Galerkin method introduced here, we take $X_0^N \subset \mathcal{H}_0^1$. A key result of the Galerkin formulation is that, over all functions in $X_0^N$, $u$ is the *best fit* approximation to $\tilde{u}$ in the *energy norm*. That is: $||u - \tilde{u}||_a \leq ||w - \tilde{u}||_a \; \forall \; w \in X_0^N$.

- The best fit property is readily demonstrated. For any $v,\, w \in \mathcal{H}_0^1$, one can show that $(v, r(w)) = a(v, w - \tilde{u})$. Defining the error $e := u - \tilde{u}$ and using the orthogonality of the residual $r(u)$, one has
$$0 = (v, r(u)) = a(v, e) \; \forall v \in X_0^N$$
That is, the error is orthogonal to the approximation space in the $a$ inner product, $e \perp_a X_0^N$. As depicted in Fig. 1, $u$ is the closest element of $X_0^N$ to $\tilde{u}$.

- For $u \in X_0^N$, and $v \in Y_0^N$ we refer to $X_0^N$ as the *trial space* and $Y_0^N$ as the test space. Provided that the cardinalities of $X_0^N$ and $Y_0^N$ are equal, the spaces need not be the same. In particular, if one chooses $Y_0^N = span\{\delta(x - x_1), \delta(x - x_2), \ldots, \delta(x - x_n)\}$ one recovers the *strong form* in which (8) is satisfied pointwise. The Galerkin statement (13) is often referred to as the *weak form*, the variational form, or the weighted residual form.

- The variational form (13) leads to symmetric positive definite system matrices, even for more general Neumann or Robin boundary conditions, which is not generally the case for finite difference methods.

**Deriving a System of Equations**

We develop (13) into a discrete system appropriate for computation by inserting the expansions $v = \sum_i v_i \phi_i$ and $u = \sum_j u_j \phi_j$ into the integrand on the left of (13) to yield

$$\int_\Omega \nabla \left( \sum_{i=1}^{n} v_i \phi_i(\mathbf{x}) \right) \cdot \nabla \left( \sum_{j=1}^{n} u_j \phi_j(\mathbf{x}) \right) dV \quad = \quad \sum_{i=1}^{n} \sum_{j=1}^{n} v_i \left( \int_\Omega \nabla \phi_i(\mathbf{x}) \cdot \nabla \phi_j(\mathbf{x}) \, dV \right) u_j \quad (14)$$

Equating (14) to the right side of (13) and defining

$$A_{ij} \quad := \quad \int_\Omega \nabla \phi_i(\mathbf{x}) \cdot \nabla \phi_j(\mathbf{x}) \, dV \quad (15)$$

$$b_i \quad := \quad \int_\Omega \phi_i(\mathbf{x}) f \, dV \quad (16)$$

$$\underline{v} \quad := \quad (v_1, v_2, \ldots, v_n)^T \quad (17)$$

$$\underline{u} \quad := \quad (u_1, u_2, \ldots, u_n)^T \quad (18)$$

the discrete Galkerin formulation becomes, *Find* $\underline{u} \in \mathbb{R}^n$ *such that*

$$\sum_{i=1}^{n} \sum_{j=1}^{n} v_i A_{ij} u_j =: \underline{v}^T A \underline{u} \; = \; \underline{v}^T \underline{b} \qquad \forall \underline{v} \in \mathbb{R}^n, \quad (19)$$

or, equivalently:

$$A\underline{u} \quad = \quad \underline{b} \, . \quad (20)$$

This is the system to be solved for the basis coefficients $u_i$. It is easy to show that $A$ is symmetric positive definite ($\underline{x}^T A \underline{x} > 0 \;\; \forall \underline{x} \neq 0$) and therefore invertible. The *conditioning* of $A$, however, is not guaranteed to be amenable to finite-precision computation unless some care is exercised in choosing the basis for approximation. Normally, this amounts merely to finding $\phi_i$s that are nearly orthogonal or, more to the point, far from being linearly dependent. We discuss good and bad basis choices shortly. (Generally speaking, one can expect to lose $\approx \log_{10} \kappa(A)$ digits due to round-off effects when solving (20).)

**Extensions**

Once the requisite properties of the trial/test spaces are identified, the Galerkin scheme is relatively straightforward to derive. One formally generates the system matrix $A$ with right hand side $\underline{b}$ and then solves for the vector of basis coefficients $\underline{u}$. Extensions of the Galerkin method to more complex systems of equations is also straightforward. For the example of the reaction-convection-diffusion equation, $-\nu \nabla^2 u + \mathbf{c} \cdot \nabla u + \alpha^2 u = f$, the procedure outlined above leads to

$$\nu A\underline{u} + C\underline{u} + \alpha^2 B\underline{u} \quad = \quad \underline{b} \, , \quad (21)$$

with $C_{ij} := \int \phi_i \, \mathbf{c} \cdot \nabla \phi_j \, dV$ and $B_{ij} := \int \phi_i \, \phi_j \, dV$. We refer to $A$ as the stiffness matrix, $B$ the mass matrix, and $C$ the convection operator. For the convection dominated case (i.e., $\nu$ small, $\alpha = 0$), one must be judicious in the choice of trial and test spaces. Another important extension is the treatment of boundary conditions other than the homogeneous Dirichlet conditions ($u = 0$ on $\partial\Omega$) considered so far. In addition, development of an *efficient* procedure requires attention to the details of the implementation. Before considering these extensions and details, we introduce some typical examples of bases for $X_0^N$.

## 1D Examples

We consider a few examples of 1D basis functions. In addition to the trial/test spaces and associated bases, we will introduce a means to compute the integrals associated with the systems matrices, $A$, $B$, and $C$. We have the option of using exact integration or inexact quadrature. In the latter case, we are effectively introducing an additional discretization error and must be mindful of the potential consequences.

Basis functions (in any space dimension $d$) come in essentially two forms, *nodal* and *modal*. Nodal basis functions are known as Lagrangian interpolants and have the property that the basis coefficients $u_i$ are also *function values* at distinct points $\mathbf{x}_i$. From the definition of $u(\mathbf{x})$, one has:

$$u(\mathbf{x}_i) \;:=\; \sum_{j=1}^{n} u_j \phi_j(\mathbf{x}_i) \;=\; u_i, \tag{22}$$

where the last equality follows from the nodal basis property. Because (22) holds for all $u_j$, Lagrangian bases must satisfy $\phi_j(\mathbf{x}_i) = \delta_{ij}$, where $\delta_{ij}$ is the Kronecker delta function that is 1 when $i = j$ and 0 otherwise. A distinct advantage of Lagrangian bases is that it is relatively simple to satisfy $u = 0$ on $\partial\Omega$—one simply excludes $\phi_i$ from the basis set for all $\mathbf{x}_i \in \partial\Omega$. (This statement is not strictly sufficient for $d > 1$, but turns out to be correct for most of the commonly used bases.)

#### Lagrange Polynomials

Figure 2 shows Lagrange polynomial interpolants of degree $N$ for two sets of nodal points $\{x_i\}$. (For these bases with Dirichlet conditions at $x = \pm 1$, we have $n = N - 1$.) Both choices lead to mathematically equivalent bases for $X_0^N \subset \mathbb{P}_N$, the space of all polynomials of degree $\leq N$. The first set of points, however, is uniformly distributed and leads to an unstable (*ill-conditioned*) formulation. The basis functions become wildly oscillatory for $N > 7$, resulting in very large values for $\phi_i'$, particularly near the domain endpoints. The second set is based on the Gauss-Lobatto-Legendre (GLL) quadrature points. For $\Omega = [-1, 1]$, the GLL points are $x_i = \xi_i$, the roots of $(1 - x^2)P_N'(x)$, where $P_N$ is the $N$th-order Legendre polynomial. We see that the GLL-based interpolants $\phi_i$ are maximal at $x = x_i$ and rapidly diminish for $|x - x_i| > 0$. Unlike the uniform points, the GLL points lead to a stable formulation. As shown in Fig. 4, the condition number of the 1D stiffness matrix grows exponentially with $N$ in the uniform case and only as $O(N^3)$ when using GLL-based cardinal points.

Stability similarly holds for almost any set of points based on the roots of classic orthogonal polynomials. One example is the set of Gauss-Lobatto-Chebyshev (GLC) points, which have the closed-form definition $\xi_j^C = -\cos(\pi j/N)$. As with other orthogonal polynomials, the GLC points lead to a high-order quadrature formula for integrals of the form $\mathcal{I}(p) := \int w(x)\, p(x)\, dx$, for a particular weight function $w(x) \geq 0$. Only the Legendre points, however, apply for the case $w(x) \equiv 1$, which is requisite for a symmetric stiffness matrix $A$. Because the GLC points are uniformly spaced on the circle, they have many symmetries that can be applied recursively to develop a fast Chebyshev transform (FCT) that allows one to change between nodal and modal bases in $O(N \log N)$ operations, a property that is unique to Chebyshev bases for $\mathbb{P}_N$. For large $N$ ($> 50$, say) the FCT can be much faster than the matrix-vector product approach (see (28) below), which requires $\sim 2N^2$ operations. In the large-$N$ limit it thus may be worthwhile to give up symmetry of $A$ in favor of speed. (In higher space dimensions, the number of memory accesses

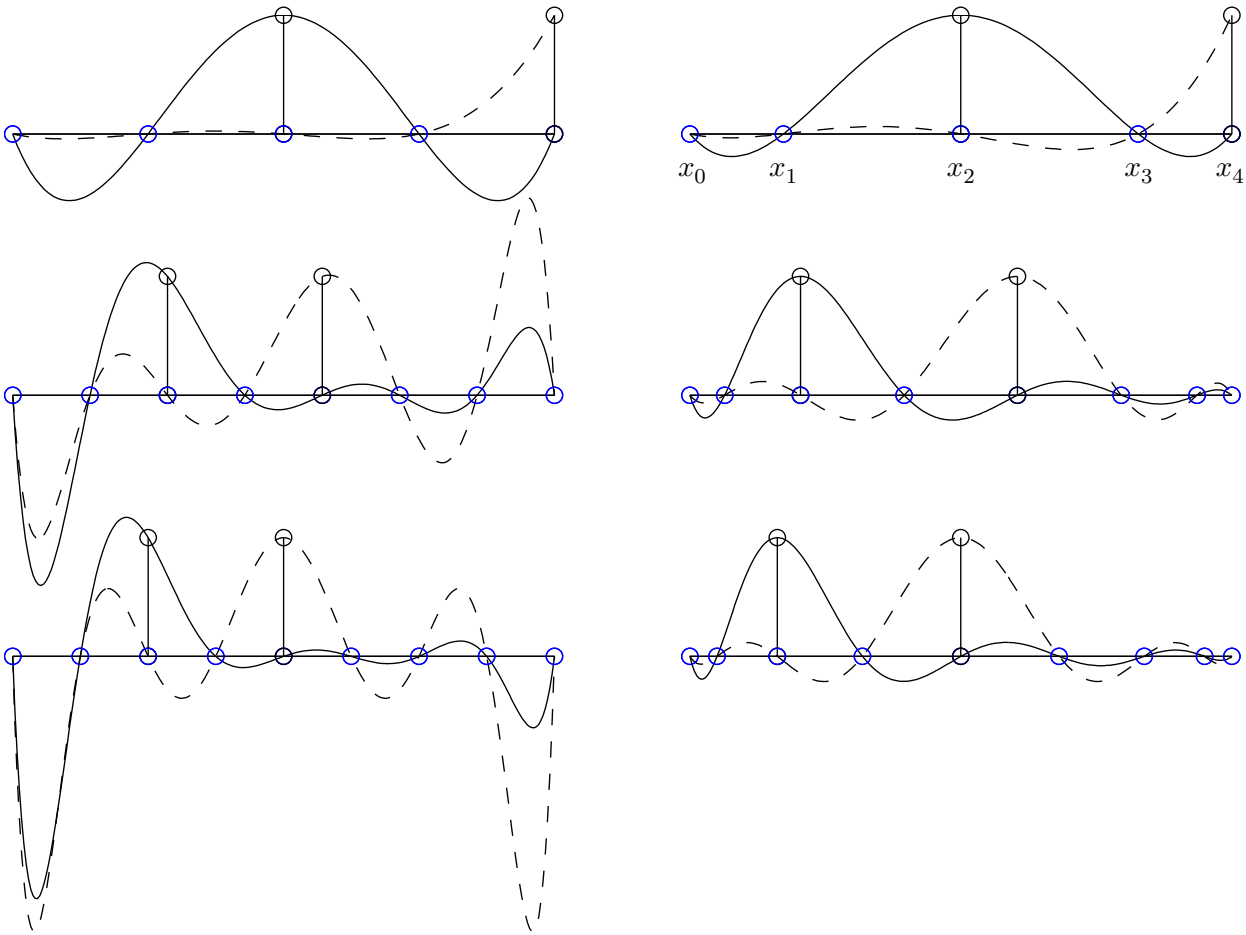


Figure 2: Lagrange polynomial basis functions $\phi_2(x)$ ( — ) and $\phi_4(x)$ ( - - ) for uniformly distributed points (left) and Gauss-Lobatto-Legendre points (right) with (from top to bottom) $N$=4, 7, and 8. Note the significant overshoot for the uniform cases with large $N$. For $N \geq 8$, the uniform distribution even results in interpolants with negative quadrature weights.

is $\sim N^d$ for both approaches, and the work is respectively $O(N^d \log N)$ and $\sim 2N^d \cdot N$, with the latter cast as fast *matrix-matrix* products.)

The GLL points have the particularly attractive property of allowing efficient *quadrature* (numerical integration). In particular, for any $p(x) \in \mathbb{P}_{2N-1}$,

$$\int_{-1}^{1} p(x)\,dx \;\equiv\; \sum_{k=0}^{N} \rho_k p(\xi_k), \tag{23}$$

with quadrature weights given by

$$\rho_k \;=\; \int_{-1}^{1} \phi_k(x)\,dx. \tag{24}$$

Thus, we have the remarkable result that any polynomial of up to degree $2N-1$ can be integrated *exactly* (to machine precision) by evaluating it at only $N+1$ points. This result is of particular relevance in computing the entries of the stiffness matrix $A$. For the 1D case on $[-1,1]$ one has:

$$\begin{aligned} A_{ij} \;:=\; \int_{-1}^{1} \phi_i'(x)\phi_j'(x)\,dx \quad &\equiv \quad \sum_{k=0}^{N} \rho_k \phi_i'(\xi_k)\phi_j'(\xi_k) \\ &= \quad \sum_{k=0}^{N} \widehat{D}_{ki}\tilde{B}_{kk}\widehat{D}_{kj}, \end{aligned} \tag{25}$$

or

$$A \quad = \quad \widehat{D}^T \tilde{B} \widehat{D}. \tag{26}$$

Here, we have introduced the 1D derivative matrix $\widehat{D}_{ki} := \phi_i'(\xi_k)$ and 1D diagonal mass matrix $\tilde{B}_{ki} := \delta_{ki}\rho_i$. Note that the integrand in (25) is of degree $2N-2$ and (26) is therefore exact. (QUIZ: What are the dimensions of $A$ and $\tilde{B}$ in (26)?)

The nodal bases introduced above are *global* in that they extend over the entire domain. For historical reasons, methods using these bases are referred to as spectral methods. They are characterized by rapid (exponential) convergence, with $||u - \tilde{u}|| \sim \sigma^N$ with $0 < \sigma < 1$, for $\tilde{u}$ sufficiently regular (differentiable).

**Compactly Supported Bases**

As an alternative to global bases, on can consider *local* bases, such as the space of piecewise polynomials illustrated by the piecewise linear and quadratic examples shown in Fig. 3. These are the standard basis functions for the finite element method (FEM) and, at higher orders, the spectral element method (SEM). These bases have the advantage of allowing a flexible distribution of gridpoints (element sizes) to capture boundary layers, shocks, and so forth, and moreover lead to *sparse* system matrices. The FEM/SEM basis is said to have *compact* or *local support*. That is, individual basis functions are nonzero only over a small subset of $\Omega$. Any two functions $\phi_i$ and $\phi_j$ that do not have intersecting support will lead to a zero in the *i-j* entry in all of the system matrices. Consider the system matrix (15) for the linear ($N$=1) basis functions of Fig. 3. It is clear that the product $q(x) := \phi_i'(x)\phi_j'(x)$ will vanish unless $x_i$ and $x_j$ are either coincident or adjacent.

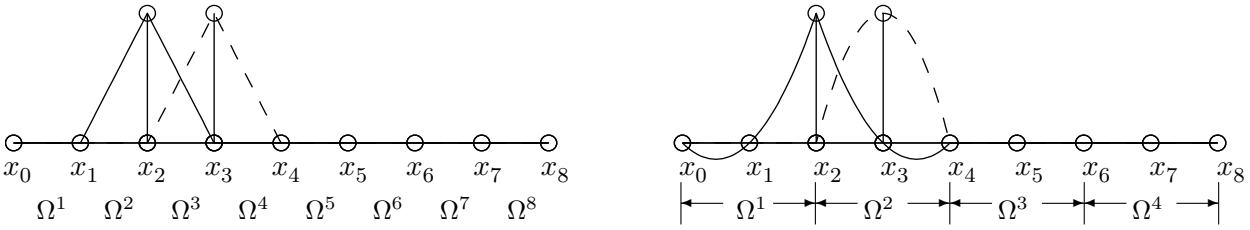


Figure 3: Examples of one-dimensional piecewise linear (left) and piecewise quadratic (right) Lagrangian basis functions, $\phi_2(x)$ and $\phi_3(x)$, with associated element support, $\Omega^e$, $e = 1, \ldots, E$.

Hence, $A_{ij} = 0$ whenever $|i-j| > 1$. That is, $A$ is a tridiagonal matrix. Such systems can be solved in $O(n)$ time and are hence nominally optimal (in 1D).

For the linear basis of Fig. 3, the convergence rate, $||u - \tilde{u}|| = O(n^{-2})$, is *much* slower than for the spectral case and one may need to take a much larger value of $n$ to realize the desired accuracy. For this reason, one often considers high-order piecewise polynomial bases as illustrated, for example, by quadratic basis functions in Fig. 3 (right). Such bases are used in spectral element and $p$-type finite element methods. Because polynomials do a much better job of approximating smooth curvy functions than their piecewise linear counterparts, we may anticipate more rapid convergence. In general, for piecewise polynomial expansions of order $N$ on each of $E$ elements, one can expect convergence rates scaling as $\sim (CE)^{-(N\pm1)}$. Details in the convergence rate vary according to the problem at hand, the chosen norm, etc. The slight increase in work associated with higher order $N$ is usually more than offset by the gain in accuracy, and high-order methods are finding widespread application in modern simulation codes.

### Modal Bases

The second class of basis functions of interest are *modal* bases. These bases do not satisfy the Lagrangian condition $\phi_i(\mathbf{x}_j) = \delta_{ij}$ but generally are designed to have other desirable properties such as orthogonality with respect to a particular inner product (thus yielding diagonal operators) or ease of manipulation. 1D examples include Fourier bases consisting of sines and cosines, e.g.,

$$X^N = \text{span}\{\cos 0x,\, \sin \pi x,\, \cos \pi x, \ldots, \sin N\pi x,\, \cos N\pi x\}, \tag{27}$$

which are optimal for constant coefficient operators on periodic domains. (Q: What is $n$ in this case?) It is common to denote modal basis coefficients with a circumflex (hat) in order to indicate that they do *not* represent the value of a function at a point, e.g., $u(x) = \sum_k \hat{u}_k \phi_k(x)$. For polynomial spaces, one can consider simple combinations of the monomials, say, $\phi_k = x^{k+1} - x^{k-1}$, $k = 1, \ldots, n$, which satisfy the homogeneous conditions $\phi_i(\pm 1) = 0$ and are easy to differentiate and integrate. The conditioning of the resultant systems, however, scales exponentially with $N$ and for even modest values of $N$ the systems become practically uninvertible in finite precision arithmetic. Modal bases for $\mathbb{P}_N$ are thus typically based on orthogonal polynomials or combinations thereof. A popular choice is $\phi_k = P_{k+1} - P_{k-1}$, $k = 1, \ldots, n$ where $P_k$ is the Legendre polynomial of degree $k$. This basis consists of a set of "bubble functions" that vanish at the domain endpoints ($\pm 1$). Like their Fourier counterparts, the bubble functions become more and more oscillatory with increasing $k$ and also lead to a diagonal stiffness matrix in 1D (VERIFY).

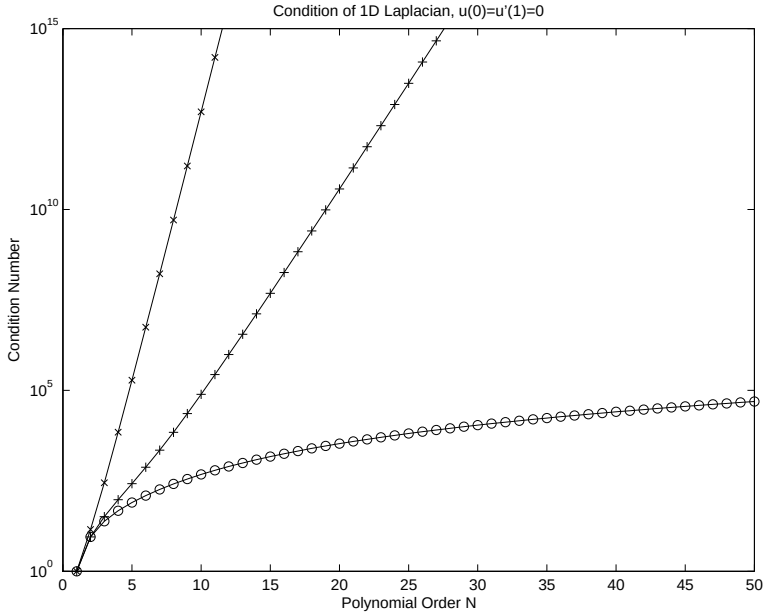


Figure 4: Condition numbers for 1D discrete Laplacian on [0,1] with $u(0) = u'(1) = 0$ using single-domain polynomial expansions of order $N$. Lagrangian nodal bases on Gauss-Lobatto-Legendre points: ○—○; nodal bases on uniform points: +—+; modal bases with monomial ($x^k$) expansions: ×—×. The condition number for the GLL-based nodal expansions scales as $N^3$.

We reiterate that either modal or nodal bases cover the *same approximation space*, $X_0^N$. Only the representation differs and it easy to change from one representation to the other. If $\underline{\hat{u}} = (\hat{u}_1, \ldots, \hat{u}_n)^T$ represents a set of modal basis coefficients, it is clear that nodal values of $u$ are given by $u_i := u(x_i) = \sum_k \hat{u}_k \phi_k(x_i)$. In matrix form,

$$\underline{u} = V\underline{\hat{u}}, \qquad V_{ik} := \phi_k(x_i). \tag{28}$$

$V$ is known as the Vandermonde matrix. For a well-chosen modal basis, the *modal-to-nodal* conversion (28) is stable for all $x_i \in [-1, 1]$. Assuming that the $x_i$ are distinct, modal coefficients can be determined from a set of nodal values through the inverse, $\underline{\hat{u}} = V^{-1}\underline{u}$. For $V^{-1}$ to be well-conditioned the nodal points must be suitably chosen, e.g., GLL or GLC points rather than uniform. Otherwise, round-off errors lead to loss of information in the modal-to-nodal transform. Given this equivalence of bases, it is clear that one can also consider element-based modal expansions to represent piecewise polynomial approximation spaces.

**Condition Number Example**

The condition number of the 1D stiffness matrix for the problem $-u''(x) = f(x)$, $u(0) = u'(1) = 0$ is shown in Fig. 4 for $N = 1$ to 50. Note that all precision is lost for the monomial ($x^k$) modal basis by $N = 10$ and for the uniformly-distributed modal basis by $N$=20. The matlab code for generating these plots is given below. Note that `SEMhat.m` generates the standard points, weights, and 1D stiffness and mass matrices, while unihat.m generates uniformly distributed points on [-1,1] and the associated derivative matrix, `Du`.

```
% bad1d.m
for N=1:50;

   [Ah,Bh,Ch,Dh,zh,wh] = SEMhat(N);  % Standard GLL points
   [Au,Bu,Cu,Du,zu,wu] = unihat(N);  % Uniform points

   No = N+2; % Overintgration
   [Ao,Bo,Co,Do,zo,wo] = SEMhat(No); J = interp_mat(zo,zu);

   Bu = J'*Bo*J; Au = Du'*Bu*Du;     % Full mass and stiffness matrices

   n=size(Ah,1);
   Ah=Ah(1:n-1,1:n-1);  % Dirichlet at x=1; Neumann at x=-1;
   Au=Au(1:n-1,1:n-1);

   NN(N)=N; cu(N)=cond(Au); cg(N)=cond(Ah);

end;


nmax=25; c=zeros(nmax,1); N=zeros(nmax,1);
for n=1:25; A=zeros(n,n);
   for i=1:n; for j=1:n;
     A(i,j) = (i*j)/(i+j-1);   % Stiffness matrix for phi_i = x^i on (0,1]
   end; end;
   N(n) = n; c(n) = cond(A);
end;

semilogy(NN,cu,'r+',NN,cu,'r-',NN,cg,'bo',NN,cg,'b-',N,c,'kx',N,c,'k-')
axis([0 50 1 1.e15]); title('Condition of 1D Laplacian, u(0)=u''(1)=0');
xlabel('Polynomial Order N'); ylabel('Condition Number');
print -deps 'bad1d.ps'
```

# 4 Higher Space Dimensions

There are many options for generating basis functions in higher space dimensions. The finite element method proceeds as in the 1D case, using compactly-supported basis functions that are nonzero only a small subset of the subdomains (elements) whose union composes the entire domain. Elements are typically either $d$-dimensional simplices (e.g., triangles for $d$=2 and tetrahedra for $d$=3) or tensor products of the 1D base element, that is, $\widehat{\Omega} := [-1,1]^d$, (e.g. quadrilaterals for $d$=2 and hexahedra– curvilinear bricks– for $d$=3). An excellent discussion of the development of high-order nodal and modal bases for simplices is given in the recent Springer volume by Hesthaven and Warburton. Here, we turn our attention to the latter approach, which forms the foundation of the spectral element method. Specifically, we will consider the case of a single subdomain where $\Omega = \widehat{\Omega} = [-1,1]^2$, which corresponds of course to a global spectral method. The spectral element method derives from this base construct by employing multiple subdomains and using preconditioned iterative methods to solve the unstructured linear systems that arise.

If the domain is a $d$-dimensional box, it is often attractive to use a tensor-product expansion

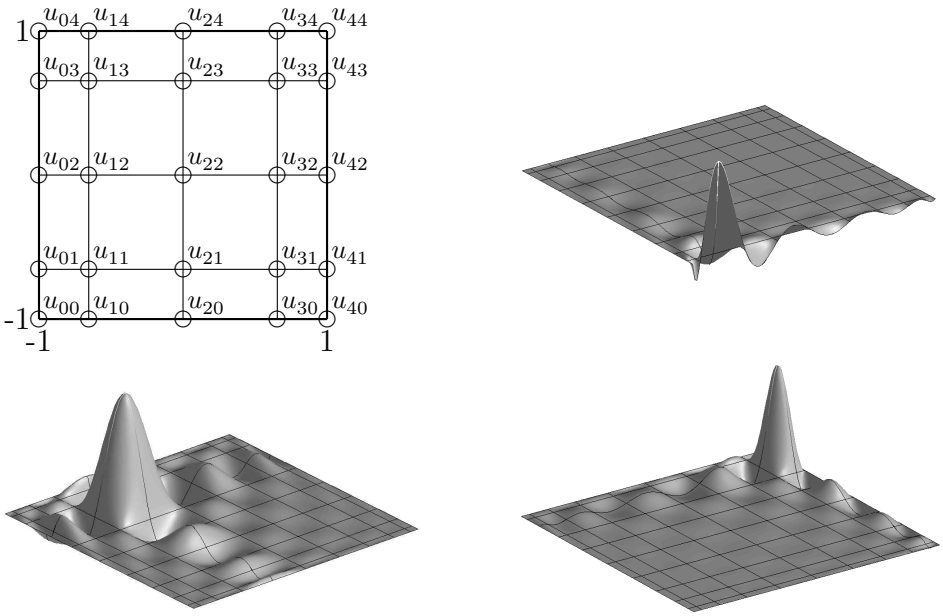


Figure 5: Clockwise from upper left: GLL nodal point distribution on $\widehat{\Omega}$ for $M = N = 4$, and Lagrangian basis functions for $M = N$=10: $h_{10}h_2$, $h_2h_9$, and $h_3h_4$.

of one-dimensional functions. One could use, for example, any of the one-dimensional bases of the previous section, including the nodal/modal, FEM/SEM or spectral. For our purposes, we'll consider stable 1D Lagrange polynomials, denoted, say, by $h_i(x)$, $i = 0, \ldots, N$, based on the GLL points on $[-1, 1]$. The corresponding 2D basis function on $\Omega := [-1, 1]^2$ would take the form

$$\phi_k(x, y) := h_i(x)h_j(y), \tag{29}$$

with $k := i + (N + 1) * j$. In principle, everything follows as before. One simply manipulates the basis functions as required to evaluate the system matrices such as defined in (25)–(26). In the section below we demonstrate some of the important savings that derive from the tensor-product structure of (29) that allow straightforward development of efficient 2D and 3D simulation codes.

# 5 Tensor Products

To begin this section we introduce a number of basic properties of matrices based on tensor-product forms. Such matrices frequently arise in the numerical solution of PDEs. Their importance for the development of fast Poisson solvers was recognized in an early paper by Lynch et al. [**?**]. In 1980, Orszag [**?**] pointed out that tensor-product forms were the foundation for efficient implementation of spectral methods.

The relevance of tensor-product matrices to multidimensional discretizations is illustrated by

considering a tensor-product polynomial on the reference domain, $(x, y) \in \widehat{\Omega} := [-1, 1]^2$:

$$u(x, y) = \sum_{i=0}^{M} \sum_{j=0}^{N} u_{ij} h_{M,i}(x) h_{N,j}(y) \,. \tag{30}$$

Here, $h_{M,i}$ (resp., $h_{N,j}$) is the Lagrangian interpolant of degree $M$ (resp., $N$) based on Legendre polynomials introduced in (**??**). Consequently, the basis coefficients, $u_{ij}$, are also nodal values of $u$ on the tensor product of Gauss-Lobatto-Legendre (GLL) quadrature points, as shown in Fig. 5. Useful vector representations of the coefficients are denoted by

$$\underline{u} \;:=\; (u_1, u_2, \ldots, u_l, \ldots, u_{\mathcal{N}})^T \;:=\; \Big(u_{00}, u_{10}, \ldots, u_{ij}, \ldots, u_{MN}\Big)^T, \tag{31}$$

where $\mathcal{N} = (M+1)(N+1)$ is the number of basis coefficients, and the mapping $l = 1+i+(M+1)j$ translates to standard vector form the two-index coefficient representation, with the leading index advancing most rapidly. This is referred to as the natural, or lexicographical, ordering and is used throughout this text.

The tensor-product form (30) allows significant simplifications in the application of linear operators to $u$. For example, suppose $w_{pq}$ is to represent the $x$-derivative of $u$ at the GLL points, $(\xi_{M,p}, \xi_{N,q})$, $p, q \in \{0, \ldots, M\} \times \{0, \ldots, N\}$,

$$\begin{aligned} w_{pq} \;:=\; \frac{\partial u}{\partial x}(\xi_{M,p}, \xi_{N,q}) \quad &= \quad \sum_{i=0}^{M} \sum_{j=0}^{N} u_{ij} h'_{M,i}(\xi_{M,p}) h_{N,j}(\xi_{N,q}) \\ &= \quad \sum_{i=0}^{M} u_{iq} h'_{M,i}(\xi_{M,p}). \end{aligned} \tag{32}$$

Expressed as a matrix-vector product, (32) reads

$$\underline{w} \;=\; D_x \underline{u} \quad := \quad \begin{bmatrix} \widehat{D}_x & & & \\ & \widehat{D}_x & & \\ & & \ddots & \\ & & & \widehat{D}_x \end{bmatrix} \begin{pmatrix} u_{00} \\ u_{10} \\ \vdots \\ u_{MN} \end{pmatrix}. \tag{33}$$

Here, $\widehat{D}_x$ is the one-dimensional derivative matrix (26) associated with the $(M+1)$ GLL points on the reference interval $[-1, 1]$ and is applied to *each row*, $(u_{0j}, u_{1j}, \ldots, u_{Mj})^T$, $j = 0, \ldots, N$, in the computational grid of Fig. 5. To compute derivatives with respect to $y$, one applies the corresponding matrix $\widehat{D}_y$ to *each column*. This latter case does not yield a simple block-diagonal matrix because of the ordering of the coefficients in $\underline{u}$. Nonetheless, $D_x$ and $D_y$ are conveniently expressed as $D_x = (I \otimes \widehat{D}_x)$ and $D_y = (\widehat{D}_y \otimes I)$, where $\otimes$ indicates the tensor (or Kronecker) product, which we now introduce.

Let $A$ and $B$ be $k \times l$ and $m \times n$ matrices, respectively, and consider the $km \times ln$ matrix C, given in block form as

$$C \quad := \quad \begin{pmatrix} a_{11}B & a_{12}B & \ldots & a_{1l}B \\ a_{21}B & a_{22}B & \ldots & a_{2l}B \\ \vdots & \vdots & & \vdots \\ a_{k1}B & a_{k2}B & \ldots & a_{kl}B \end{pmatrix}. \tag{34}$$

$C$ is said to be the tensor (or Kronecker) product of $A$ and $B$, denoted as

$$C \quad = \quad A \otimes B. \tag{35}$$

From the definition (34) one can easily verify that an entry $c_{ij}$ $(1 \le i \le km,\ 1 \le j \le ln)$ in $C$ is equal to

$$c_{ij} \ = \ a_{pq}\, b_{rs},$$

where the index couples of $(pq)$ and $(rs)$ satisfy the relationships

$$i \ = \ r + (p-1)m, \qquad\qquad j \ = \ s + (q-1)n.$$

We will occasionally also refer to $c_{ij}$ as $c_{rp,sq}$.

Using the standard definition of matrix multiplication, one may show that

$$(A \otimes B)\,(F \otimes G) \quad = \quad (AF \otimes BG)\,, \tag{36}$$

for rectangular matrices $A$, $B$, $F$, and $G$ appropriately dimensioned such that the products $AF$ and $BG$ are well defined. It follows that if $A$ and $B$ are square invertible matrices of orders $m$ and $n$, respectively, then

$$(A^{-1} \otimes B^{-1})\,(A \otimes B) \quad = \quad (I \otimes I)\,, \tag{37}$$

where $I$ denotes the identity matrix of appropriate order. Thus, the inverse of $(A \otimes B)$ is $(A^{-1} \otimes B^{-1})$.

Suppose $A$ and $B$ are diagonalizable square matrices of order $m$ and $n$, respectively, with similarity transformations

$$A = S\Lambda S^{-1}\,, \qquad\qquad B = T\mathcal{M}T^{-1}, \tag{38}$$

where the columns of the matrices $S$ and $T$ are the eigenvectors of $A$ and $B$, and the entries in the diagonal matrices $\Lambda$ and $\mathcal{M}$ are the respective eigenvalues. Then

$$(S^{-1} \otimes T^{-1})(A \otimes B)(S \otimes T) \quad = \quad (\Lambda \otimes \mathcal{M})\,.$$

Note that the tensor product of two diagonal matrices is also diagonal. Thus the diagonalization of $C$ in (35) may be obtained from the diagonalization of the two much smaller systems, $A$ and $B$.

The following form frequently arises in finite difference discretization of two-dimensional boundary value problems:

$$C \quad = \quad (A \otimes I \ + \ I \otimes B)$$

(e.g., when $A$ and $B$ represent one-dimensional difference operators). Here the inverse is not so easily obtained as in (37). However, if $A$ and $B$ are diagonalizable as in (38), then the diagonalization of $C$ is

$$C \quad = \quad (S \otimes T)\,(\Lambda \otimes I \ + \ I \otimes \mathcal{M})\,(S^{-1} \otimes T^{-1})\,, \tag{39}$$

from which it follows that

$$C^{-1} \quad = \quad (S \otimes T)\,(\Lambda \otimes I \;+\; I \otimes \mathcal{M})^{-1}\,(S^{-1} \otimes T^{-1})\,. \tag{40}$$

Note that $(\Lambda \otimes I \;+\; I \otimes \mathcal{M})$ is diagonal and therefore trivially inverted.

Weighted residual techniques frequently give rise to systems of a more general form, for example,

$$C \quad = \quad (B_y \otimes A_x \;+\; A_y \otimes B_x)\;, \tag{41}$$

where $A_*$ and $B_*$ are the one-dimensional stiffness and mass matrices associated with their respective spatial directions. Here the appropriate similarity transformation comes from a generalized eigenvalue problem of the form

$$A_* \underline{s}_i \quad = \quad \lambda_i B_* \underline{s}_i.$$

If $A_*$ is symmetric and $B_*$ is symmetric positive definite, then there exists a complete set of eigenvectors that are orthonormal with respect to the $B_*$ inner product, that is, $\underline{s}_i^T B_* \underline{s}_j \;=\; \delta_{ij}$. If $\Lambda = diag(\lambda_i)$ is the diagonal matrix of eigenvalues and $S = (\underline{s}_1, \ldots, \underline{s}_N)$ the corresponding matrix of eigenvectors, then the following similarity transformation holds:

$$S^T A_* S = \Lambda\,, \qquad S^T B_* S = I\,. \tag{42}$$

Suppose there exist matrices $\Lambda_x$ and $S_x$ such that (42) is satisfied for $A_x$ and $B_x$, and similarly for the $y$ operators. Then the inverse of $C$ (41) is given by

$$C^{-1} \quad = \quad (S_y \otimes S_x)\,(I \otimes \Lambda_x \;+\; \Lambda_y \otimes I)^{-1}\,(S_y^T \otimes S_x^T)\,. \tag{43}$$

The preceding tensor-product factorizations readily extend to more dimensions. Rather than presenting the general form, we merely illustrate the procedure by giving the result for the three-dimensional case. As we will demonstrate in the next section, weighted residual techniques for three-dimensional boundary value problems frequently give rise to system matrices of the form

$$C = (B_z \otimes B_y \otimes A_x + B_z \otimes A_y \otimes B_x + A_z \otimes B_y \otimes B_x)\,.$$

If one assumes that similarity transforms of the form (42) exist for each $(K_*, M_*)$ pair, then $C^{-1}$ is given by

$$C^{-1} \quad = \quad (S_z \otimes S_y \otimes S_x)\,D^{-1}(S_z^T \otimes S_y^T \otimes S_x^T)\,, \tag{44}$$

where

$$D \quad = \quad (I \otimes I \otimes \Lambda_x + I \otimes \Lambda_y \otimes I + \Lambda_z \otimes I \otimes I)$$

is the diagonal matrix containing the eigenvalues of the one-dimensional operators.

The forms (40) and (44) are referred to as fast diagonalization methods because they permit inversion of an $(N^d \times N^d)$ matrix in only $O(N^{d+1})$ operations (as the next section will show). They were proposed for the solution of finite difference problems in [**?**] and are often used in the solution of global spectral methods. (For Fourier and Chebyshev methods, the work complexity can be reduced to $O(N^d \log N)$ through the use of fast-transform methods that allow the one-dimensional operators $S_*$ and $S_*^T$ to be applied in $O(N \log N)$ operations.) Although strictly applicable only to constant coefficient problems, the fast diagonalization method has also been shown to be effective as an approximate local solver for domain decomposition based preconditioners [**?**, **?**] (see Section 7.2).

## Operator Evaluation

The preceding section illustrates several useful properties of matrices based on tensor-product forms. Essential for operator evaluation in numerical solution of PDEs is the fact that matrix-vector products of the form $\underline{v} = (A \otimes B)\underline{u}$ can be evaluated in an *order of magnitude fewer* operations than would be possible in the general case.

Suppose for ease of exposition that $A$ and $B$ are square matrices of order $n$ and $m$, respectively, and that $\underline{u}$ is an $mn$-component vector with entries denoted by a double subscript, $u_{ij}$, $i = 1, \ldots, m$, $j = 1, \ldots, n$. We associate a natural ordering of these entries with the $\hat{\imath}$th component of $\underline{u}$: $u_{\hat{\imath}} = u_{ij}$ *iff* $\hat{\imath} = i + m(j-1)$. With this convention, the product is

$$v_{ij} \quad = \quad \sum_{l=1}^{n} \sum_{k=1}^{m} a_{jl} b_{ik} u_{kl} \qquad \left\{ \begin{array}{l} i = 1, \ldots, m \\ j = 1, \ldots, n \end{array} \right. . \tag{45}$$

If $A$ and $B$ are full matrices, then $C = (A \otimes B)$ is full, and computation of $C\underline{u}$ by straightforward matrix-vector multiplication nominally requires $nm(2nm-1)$ operations, or $O(n^4)$ if $m = n$. (We denote an operation as either floating-point addition or multiplication.) In practice, however, the matrix $C$ is never explicitly formed; one evaluates only the *action* of $C$ on a vector. The usual approach is to exploit the relationship (36) and the associativity of matrix multiplication to rewrite $C\underline{u}$ as

$$C\underline{u} \quad = \quad (A \otimes I)(I \otimes B)\, \underline{u}\,. \tag{46}$$

One then computes $\underline{w} = (I \otimes B)\, \underline{u}$:

$$w_{ij} \quad = \quad \sum_{k=1}^{m} b_{ik} u_{kj}\,, \quad i = 1, \ldots, m\,,\ j = 1, \ldots, n\,, \tag{47}$$

followed by $\underline{v} = (A \otimes I)\, \underline{w}$:

$$\begin{aligned} v_{ij} \quad &= \quad \sum_{l=1}^{n} a_{jl} w_{il}\,, \quad i = 1, \ldots, m\,,\ j = 1, \ldots, n\,, \\ &= \quad \sum_{l=1}^{n} w_{il} a_{lj}^{T}\,, \quad i = 1, \ldots, m\,,\ j = 1, \ldots, n\,. \end{aligned} \tag{48}$$

The total work is only $2nm(n+m-1)$, or $O(n^3)$ if $n = m$.

The three-dimensional case follows similarly and yields even greater reduction in the number of operations. Consider the tensor product of three full $n \times n$ matrices, $C = A_z \otimes A_y \otimes A_x$. If formed explicitly, $C$ will contain $n^6$ nonzeros, and the matrix vector product $\underline{v} = C\underline{u}$ will require $2n^6$ operations. However, if $C$ is factored as $C = (A_z \otimes I \otimes I)(I \otimes A_y \otimes I)(I \otimes I \otimes A_x)$ and the matrix-vector product is carried out as above for the two-dimensional case, the total cost is only $3 \times 2n^4$ operations. In general, the rule for a matrix operator resulting from a discretization with $n$ mesh points per space dimension, in $d$-space dimensions (full coupling in *each* spatial direction), is that it should be possible to evaluate the action of an operator in only $O(n^{d+1})$ operations.

We are now in a position to establish an important result for multidimensional problems: *Tensor-product-based high-order methods become increasingly effective as the spatial dimension,*

*d, increases.* Consider an example where the number of points required to resolve a function in $d$ space dimensions to a given accuracy is $N^d$ for a high-order method and $\widehat{N}^d$ for a low-order scheme with $\sigma := N/\widehat{N} < 1$. For explicit methods and for implicit schemes with iterative solvers, the leading order cost is associated with operator evaluation, which scales as $c_h N^{d+1}$ in the high-order case and $c_l \widehat{N}^d$ in the low-order case. The work ratio is thus

$$\frac{\text{Work (high-order method)}}{\text{Work (low-order method)}} = \frac{c_h}{c_l} N \sigma^d.$$

Clearly, any high-order advantage for $d = 1$ is further increased as $d$ is increased from 1 to 3. We will see in Section **??** that $\frac{c_h}{c_l} \approx 1$; we have already seen in Table **??** that $\sigma$ can be significantly less than unity.

It is important to note that, if we view the entries of $\underline{u}$ as a matrix $U$ with $\{U\}_{ij} := u_{ij}$, then the forms (47)–(48) can be recast as *matrix-matrix* products

$$(A \otimes B)\,\underline{u} = B\,U\,A^T\,. \tag{49}$$

A similar form exists for the three-dimensional case. Because of their low memory-bandwidth requirements and high degree of parallelism, optimized matrix-matrix product routines are available that are extremely fast on modern cache-based and pipelined architectures. This feature is particularly significant for three-dimensional applications, where typically more than 90% of the operation count is expended on evaluating forms similar to (49).

**Boundary Condition Extensions**

Another important extension, which is initially somewhat difficult to understand but in fact is a natural extension of the Galerkin scheme is the incorporation of Neumann or Robin boundary conditions. We consider homogeneous Neumann conditions first. Assume that we have a mixture of Dirichlet and Neumann conditions on $\partial\Omega_D$ and $\partial\Omega_N$ respectively, such that $\partial\Omega = \partial\Omega_D \bigcup \partial\Omega_N$.